Test Cases

Centraal Overstappen Messaging Platform V4.0

**Index**

[2 Version control 3](#_Toc453854700)

[3 Tests 3](#_Toc453854701)

[3.1 Functional tests COMP receives SOAP message. 3](#_Toc453854702)

[3.1.1 testcases XML validation 3](#_Toc453854703)

[3.1.2 Testcases store message processing 6](#_Toc453854704)

[3.1.3 Testing smart validations (sunny day) 15](#_Toc453854705)

[3.1.4 Testing proces flow validation state transitions (sunny day) 16](#_Toc453854706)

[3.1.5 Testing duplicate messages (rainy day) 17](#_Toc453854707)

[3.1.6 Testing Error Codes (rainy days) 17](#_Toc453854708)

[3.1.7 Error codes – unknown how to test 18](#_Toc453854709)

[3.1.8 Scheduled processes 18](#_Toc453854710)

[3.2 Functional tests COMP sends SOAP message 19](#_Toc453854711)

[3.3 Functional tests GUI 19](#_Toc453854712)

[3.3.1 Functionality tests 20](#_Toc453854713)

[3.3.2 State transitions between GUI and SOAP 20](#_Toc453854714)

[3.3.3 Testing settings 20](#_Toc453854715)

[3.4 Performance tests 22](#_Toc453854716)

[3.5 Testing notification emails, reporting emails and error emails 22](#_Toc453854717)

# Version control

|  |  |  |  |
| --- | --- | --- | --- |
| **nr** | **Date** | **Description** | **Author** |
| 0.1 | 2016-5-24 | Initial version | Ed Zwinkels |
| 0.2 | 2016-5-27 | Added: tests for message correctness | Ed Zwinkels |
| 0.3 | 2016-5-31 | Added: numbers, corrections | Ed Zwinkels |
| 0.4 | 2016-6-17 | reviewed, aligned numbers with numbers of existing testcases, more details for screens. | Ed Zwinkels |

# Tests

## Functional tests COMP receives SOAP message.

### testcases XML validation

|  |  |
| --- | --- |
|  |  |
| Test purpose | Verify that XML is validated, so an XSD violation leads to a rejection of the message. |
| Test steps | 1. Send CTR with the condition in the testcase description, store transactionId 2. Check soap\_transactions table for successful "Received" record with the stored transactionId; 3. Wait for poller; 4. Check for raw\_soap\_xml\_record with relation to soap\_transaction record and with state "Failed" 5. Check for message\_logging record with the clc\_id for XSD-error (34 on test) 6. Check in soap\_xml\_message table for record send from 'COIN' to recipient message\_type = 'Error found' containing in the sxm\_xml\_message the dossier\_id of the sent message. |
| Conditions | Platform is available and can process messages.  Recipient is configured correctly for the soap version that is tested and has all message types configured for this soap version. |

The testcases for XML-header are valid for all message types:

Already built:

|  |  |
| --- | --- |
| **nr** | **Rainy day test cases for soapv2 and soapv3** |
|  | **XML header validations:** |
| A1.1 | missing sender block |
| A1.2 | sender sp too long |
| A1.3 | sender sp too short |
| A1.4 | empty sender sp |
| A1.5 | empty sender no |
| A1.6 | sender no too short |
| A1.7 | sender no too long |
| A1.8 | sender sp incorrect format |
| A1.9 | missing receiver SP |
| A1.10 | missing receiver block |
| A1.11 | receiver SP empty |
| A1.12 | receiver NO empty |
| A1.13 | receiver SP incorrect format |
| A1.14 | receiver NO incorrect format |
| A1.15 | empty messageid |
| A1.16 | messageid too long |
| A1.17 | missing timestamp |
| A1.18 | empty timestamp |
| A1.19 | timestamp incorrect format |

CTR validations (already built except for marked yellow)

|  |  |
| --- | --- |
| **nr** | **Rainy day CTR XML body testcases for soapv2 and soapv3** |
| A2.1 | missing dossierid |
| A2.2 | incorrect dossierid format (mismatch regexp) |
| A2.3 | dossierid too long (>40) |
| A2.4 | Dossierid too short (>8) |
| A2.5 | incorrect recipient serviceprovider (invalid characters) |
| A2.6 | two valid recipient serviceprovider headers |
| A2.7 | empty recipient serviceprovider |
| A2.8 | missing recipient serviceprovider (220-error for v2) |
| A2.9 | recipient serviceprovider too long |
| A2.10 | incorrect donor serviceprovider (invalid characters) |
| A2.11 | missing donor serviceprovider (221-error for v2) |
| A2.12 | incorrect recipient networkoperator format |
| A2.13 | incorrect donor networkoperator format |
| A2.14 | missing wishdatetime |
| A2.15 | incorrect wishdatetimeformat |
| A2.16 | missing business (v3 only) |
| A2.17 | incorrect business (v3 only) |
| A2.18 | empty business (v3 only) |
| A2.19 | empty earlytermination (v3 only) |
| A2.20 | incorrect earlytermination (v3 only) |
| A2.21 | name too long |
| A2.22 | missing name |
| A2.23 | missing address block |
| A2.24 | missing postcode |
| A2.25 | postcode incorrect format |
| A2.26 | empty postcode |
| A2.27 | missing housenumber |
| A2.28 | empty housenumber |
| A2.29 | housenumber too long |
| A2.30 | housenumber incorrect characters |
| A2.31 | housenumber extension too long |
| A2.32 | empty housenumber extension |
| A2.33 | number series block with unknown tag |
| A2.34 | empty numberseries block |
| A2.35 | numberseriesstart missing |
| A2.36 | numberseries start incorrect format |
| A2.37 | numberseries start empty |
| A2.38 | numberseries end incorrect format |
| A2.39 | numberseries end empty |
| A2.40 | validationblock empty |
| A2.41 | validationblock name empty |
| A2.42 | validationblock name missing |
| A2.43 | validationblock name too long |
| A2.44 | Validationblock two consecutivename tags |
| A2.45 | validationblock value empty |
| A2.46 | validationblock value missing |
| A2.47 | validationblock value too long |
| A2.48 | Validation block two consecutive value tags |
| A2.49 | validationblock name/value/name |
| A2.50 | note empty |
| A2.51 | note too long |

To be developed:

|  |  |
| --- | --- |
|  | **Rainy day CTRA XML body testcases for soapv2 and soapv3** |
| A3.1 | missing dossierid |
| A3.2 | incorrect dossierid format (mismatch regexp) |
| A3.3 | dossierid too long (>40) |
| A3.4 | Dossierid too short (>8) |
| A3.5 | Missing business (v3 only) |
| A3.6 | Business incorrect format (v3 only) |
| A3.7 | Missing blocking |
| A3.8 | Blocking incorrect format |
| A3.9 | Empty blocking |
| A3.10 | Wishdatetime incorrect format (pattern mismatch) |
| A3.11 | Empty wishdatetime |
| A3.12 | Blockingcode incorrect format |
| A3.13 | Empty blockingcode |
| A3.14 | Proposeddatetime incorrect format |
| A3.15 | Empty proposedatettime |
| A3.16 | Empty infrablock |
| A3.17 | Infraprovider too long |
| A3.18 | Empty infraprovider |
| A3.19 | Infratype too long |
| A3.20 | Empty infratype |
| A3.21 | Infraid too long |
| A3.22 | Empty infraid |
| A3.23 | Empty note |
| A3.24 | Note too long |

To be developed:

|  |  |
| --- | --- |
|  | **Rainy day CTP XML body testcases for soapv2 and soapv3** |
| A4.1 | missing dossierid |
| A4.2 | incorrect dossierid format (mismatch regexp) |
| A4.3 | dossierid too long (>40) |
| A4.4 | Dossierid too short (>8) |
| A4.5 | actualdatetime incorrect format (pattern mismatch) |
| A4.6 | Empty actualdatetime |
| A4.7 | Empty note |
| A4.8 | Note too long |

To be developed:

|  |  |
| --- | --- |
|  | **Rainy day CTC XML body testcases for soapv2 and soapv3** |
| A5.1 | missing dossierid |
| A5.2 | incorrect dossierid format (mismatch regexp) |
| A5.3 | dossierid too long (>40) |
| A5.4 | Dossierid too short (>8) |
| A5.5 | Empty note |
| A5.6 | Note too long |

To be developed:

|  |  |
| --- | --- |
|  | **Rainy day CTCH XML body testcases for soapv2 and soapv3** |
| A6.1 | missing dossierid |
| A6.2 | incorrect dossierid format (mismatch regexp) |
| A6.3 | dossierid too long (>40) |
| A6.4 | Dossierid too short (>8) |
| A6.5 | wishdatetime incorrect format (pattern mismatch) |
| A6.6 | Empty wishdatetime |
| A6.7 | Empty note |
| A6.8 | Note too long |

To be developed:

|  |  |
| --- | --- |
|  | **Rainy day CTCHA XML body testcases for soapv2 and soapv3** |
| A7.1 | missing dossierid | |
| A7.2 | incorrect dossierid format (mismatch regexp) | |
| A7.3 | dossierid too long (>40) | |
| A7.4 | Dossierid too short (>8) | |
| A7.5 | Missing blocking | |
| A7.6 | Blocking incorrect format | |
| A7.7 | Empty blocking | |
| A7.8 | wishdatetime incorrect format (pattern mismatch) | |
| A7.9 | Empty wishdatetime | |
| A7.10 | proposeddatetime incorrect format (pattern mismatch) | |
| A7.11 | Empty proposeddatetime | |
| A7.12 | Empty note | |
| A7.13 | Note too long | |

### Testcases store message processing

Developed for CTR.

|  |  |
| --- | --- |
|  |  |
| Test purpose | Verify that incoming messages are stored as designed |
| Test steps | 1. Send valid message with the fields in the testcase description, store transactionId 2. Check soap\_transactions table as below 3. Check for raw\_soap\_xml\_messages record as below |
| Conditions | * Valid CTR, known SOAP sender. |
| Variations | This should work for any message, so CTR, CTRA etc. and v2/v3, business should not matter. The validation for 'failed' will only work for sunny day scenarios. |

#### A.1 Store received message

Already built

Send any message to COMP with a known sender and verify that it is processed into the tables soap\_transactions and raw\_soap\_xml\_messages:

|  |  |
| --- | --- |
| **Message field**  **Soap\_transactions** | **Database field** |
| st\_transaction\_id | (key for finding the record) – must be equal to the value returned to the soap call. |
| id | No check |
| st\_send\_received | Is 'received' |
| st\_sender | filled with contents of xml\_message <sender><serviceprovider> field |
| st\_receiver | 'COIN' |
| st\_message | Filled with sent soap message |
| st\_message\_datetime | within one minute from moment of sending message? |
| st\_status | 'New' |
| st\_technical\_sender | filled with the contents of the soap sender <sch:Sender>- currently empty. |
| st\_status\_datetime | filled with a timestamp == createddate? |
| createddate | Filled with the datetime of receiving the message? |
| modifieddate | Filled with the datetime of the receiving of the message? |

|  |  |
| --- | --- |
| **Message field**  **Raw\_soap\_xml\_messages** | **Database field** |
| rsxm\_st\_id | (key) – must be equal to soap\_transactions.id for same message. |
| id | No check |
| rsxm\_xml\_message | Filled with XML message from soap\_message |
| rsxm\_status | not 'Failed' |
| createddate | Filled with the datetime of receiving the message? |
| Modifieddate | Filled with the datetime of the receiving of the message? |
| Process\_datetime | Filled with same datetime as modifieddate |

#### A.2 Store a valid CTR message

Not developed, only detailed:

Generic test case:

|  |  |
| --- | --- |
|  |  |
| Test purpose | Verify that a valid CTR that is sent to COMP is stored in the database as designed |
| Test steps | Send CTR message |
| Conditions | * request is of the correct soap version for the recipient operator; * existing and active donor operator * Unique dossierId * Unique contractid/telephone number/iban for postcode; |
| Variations: | Valid CTR messages, variations:   * Different recipient operators: v2, v3, with or without GUI, GUI only * V2 and V3; * Different donor operators: v2, v3, with or without GUI, GUI only. * Donor with and without business\_gui\_only * With and without housenr\_ext tag present * With and without timestamp; * With and without messageid; * With and without notefield * With parameters.par\_ctp\_buffering is true and false; * V3 with business = Y and business = N * V3 with no earlytermination, earlytermination = Y and earlytermination = N |

This comes basically down to verifying if X1 is built as specified.

Sunny day: processing a CTR from raw\_soap\_xml\_messages into record for contract\_termination\_dossier, dossier\_messages, dossier\_phonenumbers, dossier\_validations and soap\_xml\_records.

|  |  |
| --- | --- |
| **Database field**  **Contract\_termination \_dossier** | **Validation of message** |
| id | No check |
| ctd\_spr\_recipient\_id | Filled with id of service\_provider record that has the <recipient> value from the xml message as service\_provider.spr\_code |
| ctd\_spr\_donor\_id | Filled with id of service\_provider record that has the <donor> value from the xml message as service\_provider.spr\_code |
| ctd\_prv\_id | 1 |
| ctd\_ebc\_id | empty |
| ctd\_dossier\_status | 'Pending' |
| ctd\_process\_status | 'Request' |
| ctd\_switchid | Filled with <dossierid> from xml\_message |
| ctd\_contractid | If contractid is present in message, then stored in this field |
| ctd\_requesteddatetime | Filled with <wishdatetime> from xml\_message |
| ctd\_ctr\_wishdatetime | Filled with <wishdatetime> from xml\_message |
| ctd\_ctra\_proposeddatetime | empty |
| ctd\_emailaddresscontact | Empty (legacy field) |
| ctd\_telephonecontact | Empty (legacy field) |
| ctd\_contractinitials | 'A' (default value for dossiers created with GUI, SOAPv2 or SOAPv3) |
| ctd\_contractprefix | Empty (legacy field) |
| ctd\_contractlastname | Filled with <name> from xml\_message |
| ctd\_contractzipcode | Filled with <postcode> from xml\_message |
| ctd\_contracthousenumber | Filled with <housenr> from xml\_message |
| ctd\_contracthousenumberext | Filled with <housenr\_ext> from xml\_message |
| ctd\_connectionzipcode | Empty (legacy field) |
| ctd\_connectionhousenumber | Empty (legacy field) |
| ctd\_connectionhousenumberext | Empty (legacy field) |
| ctd\_note character | Filled with <note> |
| ctd\_ctp\_migration | False , legacy field |
| active | 'true' |
| createddate | Filled with date value close to moment of processing the message |
| modifieddate | Filled with date value close to moment of processing the message |
| ctd\_dm\_last\_msgdatetime | Filled with date value close to moment of processing the message |
| ctd\_ctp\_actualdatetime | Empty |
| ctd\_customerid | Empty (legacy field) |
| ctd\_contractcompanyname | Empty (legacy field) |
| ctd\_contractstreet | Empty (legacy field) |
| ctd\_contractcity | Empty (legacy field) |
| ctd\_contractcountry | Empty (legacy field) |
| ctd\_connectionstreet | Empty (legacy field) |
| ctd\_connectioncity | Empty (legacy field) |
| ctd\_connectioncountry | Empty (legacy field) |
| ctd\_ctp\_buffering | 'true' if parameter par\_ctp\_buffering is true |
| ctd\_npr\_id | Empty (for future extension with network\_provider) |
| ctd\_ift\_id | Empty (for future extension with infra\_type) |
| ctd\_ctr\_dt\_id | 2 if <business>Y</business> present on xml message (corresponds with dossier\_type.dt\_name 'Business'), 1 in all other cases (corresponds with dossier\_type.dt\_name 'Consumer'), |
| ctd\_ctra\_dt\_id | Empty |
| ctd\_ctch\_wishdatetime | Empty |
| ctd\_ctcha\_proposeddatetime | Empty |
| ctd\_contract\_earlytermination | 'true' if <earlytermination>Y</earlytermination> present on xml message, 'false' in all other cases (N or not present) |
| ctd\_contractaddress | Empty (legacy field) |
| ctd\_connectionaddress | Empty (legacy field) |
| ctd\_infra\_id | Empty (legacy field) |
| ctd\_version | Filled with date value close to moment of processing the message |

A Dossier\_messages record must always be created when a CTR record is created.

|  |  |
| --- | --- |
| **Database field**  **Dossier\_messages** | **Validation of message** |
| id | No check |
| dm\_ctd\_id | Id of the related contract\_termination\_dossier record |
| dm\_ebc\_id | empty |
| dm\_mst\_id | Id of the message\_type record that has mst\_msgtype 'Request' |
| dm\_it\_id | Id of the interface\_type record that corresponds to the soapversion of the XML message |
| dm\_message\_status | 0 |
| dm\_msgidentfier | If present filled with xml message field <messageid> if present, otherwise empty |
| dm\_msgdatetime | Filled with date value close to moment of processing the message |
| dm\_datetime | Filled with xml message field <wishdatetime> |
| dm\_note | Filled with xml message field <note> if present, otherwise empty |
| dm\_iteration\_count |  |
| active | 'true' |
| createddate | Filled with date value close to moment of processing the message |
| modifieddate | Filled with date value close to moment of processing the message |

If <numberseries> is present a dossier\_phonenumbers record must be created.

|  |  |
| --- | --- |
| **database field**  **Dossier\_phonenumbers** | **Validation** |
| id | No check |
| dp\_ctd\_id | Refers to ctd record |
| dp\_numberserie\_start | <numberseries><start> |
| dp\_numberserie\_end | <numberseries><end> if presentP |
| active | 'true' |
| createddate | Filled with date value close to moment of processing the message |
| modifieddate | Filled with date value close to moment of processing the message |

If in <validationblock> a <name> 'contractid' is present a dossier\_validations record is created, if in <validationblock> a <name> 'iban' is present also a dossier\_validations record is created.

|  |  |
| --- | --- |
| **database field**  **Dossier\_validations** | **Validation** |
| id | No check |
| dv\_ctd\_id | Refers to ctd\_record |
| dv\_name | ‘contractid’ |
| dv\_value | <validationblock>  <name>contractid</name> or <name>iban</name>  <value> |
| active | 'true' |
| createddate | Filled with date value close to moment of processing the message |
| modifieddate | Filled with date value close to moment of processing the message |

A message\_logging entry should be created:

|  |  |
| --- | --- |
| **database field**  **Message\_logging** | **validation** |
| id | No validation |
| msl\_msgidentifier | <messageid> |
| msl\_timestamp | Filled with date value close to moment of processing the message |
| msl\_mail\_subject | Legacy field for email interface, empty |
| msl\_mail\_from\_address | Legacy field for email interface, empty |
| msl\_mail\_to\_address | Legacy field for email interface, empty |
| msl\_ctd\_id | Refers to contract\_termination\_request record |
| msl\_dm\_id | Refers to dossier\_messages record created |
| msl\_it\_id | Corresponds with interface type the message is sent with. |
| msl\_clc\_id | Refers to id of comp\_logging\_code record where clc\_code = ‘503’ |
| msl\_mail\_message\_id | Legacy field for email interface, empty |
| msl\_log character | Contains log entry. Starts with ‘503 - Marking message as commit success - Dossier Message Saved successfully’ |
| msl\_mail\_message\_blob | Legacy field for email interface, empty |
| msl\_soap\_message\_blob | The received XML message |
| active boolean | True |
| createddate | Filled with date value close to moment of processing the message |
| modifieddate | Filled with date value close to moment of processing the message |
| msl\_soap\_transaction\_id | Filled with transactionid of received message |
| msl\_from\_spr\_code | <sender> |
| msl\_to\_spr\_code | <receiver> |
| msl\_admin | For successful message ‘false’ |

#### A.3 Store date on sending a valid CTR message to the donor provider

Not developed, only detailed:

If donor\_serviceprovider supports SOAP check soap\_xml\_messages and soap\_xml\_transactions.

|  |  |
| --- | --- |
| **database field**  **Soap\_xml\_messages** | **validation** |
| id | No validation |
| sxm\_sender | <recipient> |
| sxm\_receiver | <donor> |
| sxm\_xml\_message\_type | ‘Request’ |
| sxm\_xml\_message | A valid XML message [how to check?] |
| sxm\_xml\_message\_datetime | Filled with date value close to moment of processing the message |
| sxm\_status | Depends upon soap listener being able to receive message and return transactionid |
| sxm\_status\_datetime | Depends on soap listener |
| sxm\_retry\_count | Depends on soap listener |
| sxm\_it\_id | Corresponds with soap-version supported by donor |
| sxm\_dt\_id | 2 if <business> = Y, 1 in other cases. |
| active | 'true' |
| createddate | Filled with date value close to moment of processing the message |
| modifieddate | Filled with date value close to moment of processing the message |

|  |  |
| --- | --- |
| **database field**  **Soap\_xml\_transactions** | **validation** |
| id | No validation |
| message\_id | refers to soap\_xml\_message record |
| retry\_count | Depends on soap listener |
| sxm\_sender | <recipient> |

#### A.4 Store a valid CTRA OK message

Not developed, only detailed:

For the valid CTRA message the Store received message should be tested as described in 3.1.2.1.

Generic test case:

|  |  |
| --- | --- |
|  |  |
| Test purpose | Verify that a valid CTRA that is sent to COMP is stored in the database as designed |
| Test steps | Send CTRA OK message |
| Conditions | * Valid CTR message is processed coming from receiver of CTRA message. * CTRA is of the correct soap version for the donor provider; * existing and active recipient and donor provider * dossierId is equal to CTR message’s dossier*d* * Unique contractid/telephone number/iban for postcode (all filled) * Wishdatetime is equal to wishdatetime from request |
| Variations: | Valid CTRA messages, variations:   * V2 and V3 if applicable for sender; * Different donor operators: v2 and/or v3, with or without GUI, GUI only. * Different recipient operators: v2 and/or v3, with or without GUI, GUI only * Recipient with and without business\_gui\_only * With and without timestamp; * With and without messageid; * With and without notefield * V3 with business = Y and business = N.   Blocking Y and N is process flow validation, so no variations in this testcase. |

This comes basically down to verifying if X1 is built as specified.

Sunny day: processing a CTRA from raw\_soap\_xml\_messages into record for contract\_termination\_dossier, dossier\_messages, dossier\_phonenumbers, dossier\_validations and soap\_xml\_records.

|  |  |
| --- | --- |
| **Database field**  **Contract\_termination \_dossier** | **Validation of message** |
| id | No validation |
| ctd\_spr\_recipient\_id | Filled with id of service\_provider record that has the <receiver> value from the xml message as service\_provider.spr\_code |
| ctd\_spr\_donor\_id | Filled with id of service\_provider record that has the <sender> value from the xml message as service\_provider.spr\_code |
| ctd\_prv\_id | 1 |
| ctd\_ebc\_id | Filled with id of error\_blocking\_code that has ebc\_code = <blockingcode> |
| ctd\_dossier\_status | 'Pending' |
| ctd\_process\_status | 'Request Answer OK' |
| ctd\_switchid | Matches <dossierid> from xml\_message |
| ctd\_contractid | No validation |
| ctd\_requesteddatetime | Date in <wishdattime> in message matches date in this field |
| ctd\_ctr\_wishdatetime | No validation, must be unchanged |
| ctd\_ctra\_proposeddatetime | empty |
| ctd\_emailaddresscontact | Empty (legacy field) |
| ctd\_telephonecontact | Empty (legacy field) |
| ctd\_contractinitials | 'A' |
| ctd\_contractprefix | Empty (legacy field) |
| ctd\_contractlastname | No validation, must be unchanged |
| ctd\_contractzipcode | No validation, must be unchanged |
| ctd\_contracthousenumber | No validation, must be unchanged |
| ctd\_contracthousenumberext | No validation, must be unchanged |
| ctd\_connectionzipcode | Empty (legacy field) |
| ctd\_connectionhousenumber | Empty (legacy field) |
| ctd\_connectionhousenumberext | Empty (legacy field) |
| ctd\_note character | No validation, must be unchanged |
| ctd\_ctp\_migration | No validation, must be unchanged |
| active | 'true' |
| createddate | No validation, must be unchanged |
| modifieddate | Filled with date value close to moment of processing the message |
| ctd\_dm\_last\_msgdatetime | Filled with date value close to moment of processing the message |
| ctd\_ctp\_actualdatetime | Empty |
| ctd\_customerid | Empty (legacy field) |
| ctd\_contractcompanyname | Empty (legacy field) |
| ctd\_contractstreet | Empty (legacy field) |
| ctd\_contractcity | Empty (legacy field) |
| ctd\_contractcountry | Empty (legacy field) |
| ctd\_connectionstreet | Empty (legacy field) |
| ctd\_connectioncity | Empty (legacy field) |
| ctd\_connectioncountry | Empty (legacy field) |
| ctd\_ctp\_buffering | 'true' if parameter par\_ctp\_buffering is true |
| ctd\_npr\_id | Empty (for future extension with network\_provider) |
| ctd\_ift\_id | Empty (for future extension with infra\_type) |
| ctd\_ctr\_dt\_id | 2 if <business>Y</business> present on xml message (corresponds with dossier\_type.dt\_name 'Business'), 1 in all other cases (corresponds with dossier\_type.dt\_name 'Consumer'), |
| ctd\_ctra\_dt\_id | 2 if <business>Y</business> present on xml message (corresponds with dossier\_type.dt\_name 'Business'), 1 in all other cases (corresponds with dossier\_type.dt\_name 'Consumer'), |
| ctd\_ctch\_wishdatetime | Empty |
| ctd\_ctcha\_proposeddatetime | Empty |
| ctd\_contract\_earlytermination | No validation, must be unchanged |
| ctd\_contractaddress | Empty (legacy field) |
| ctd\_connectionaddress | Empty (legacy field) |
| ctd\_infra\_id | Empty (legacy field) |
| ctd\_version | Filled with date value close to moment of processing the message |

A Dossier\_messages record must always be created when a CTR record is created.

|  |  |
| --- | --- |
| **Database field**  **Dossier\_messages** | **Validation of message** |
| id | No check |
| dm\_ctd\_id | Id of the related contract\_termination\_dossier record |
| dm\_ebc\_id | Filled with id of error\_blocking\_code that has ebc\_code = <blockingcode> |
| dm\_mst\_id | Id of the message\_type record that has mst\_msgtype 'Request Answer' |
| dm\_it\_id | Id of the interface\_type record that corresponds to the soapversion of the XML message |
| dm\_message\_status | 0 |
| dm\_msgidentfier | If present filled with xml message field <messageid> if present, otherwise empty |
| dm\_msgdatetime | Filled with date value close to moment of processing the message |
| dm\_datetime | Filled with xml message field <wishdatetime> |
| dm\_note | Filled with xml message field <note> if present, otherwise empty |
| dm\_iteration\_count |  |
| active | 'true' |
| createddate | Filled with date value close to moment of processing the message |
| modifieddate | Filled with date value close to moment of processing the message |

A message\_logging entry should be created:

|  |  |
| --- | --- |
| **database field**  **Message\_logging** | **validation** |
| id | No validation |
| msl\_msgidentifier | <messageid> |
| msl\_timestamp | Filled with date value close to moment of processing the message |
| msl\_mail\_subject | Legacy field for email interface, empty |
| msl\_mail\_from\_address | Legacy field for email interface, empty |
| msl\_mail\_to\_address | Legacy field for email interface, empty |
| msl\_ctd\_id | Refers to contract\_termination\_request record |
| msl\_dm\_id | Refers to dossier\_messages record created |
| msl\_it\_id | Corresponds with interface type the message is sent with. |
| msl\_clc\_id | Refers to id of comp\_logging\_code record where clc\_code = ‘503’ |
| msl\_mail\_message\_id | Legacy field for email interface, empty |
| msl\_log character | Contains log entry. Starts with ‘503 - Marking message as commit success - Dossier Message Saved successfully’ |
| msl\_mail\_message\_blob | Legacy field for email interface, empty |
| msl\_soap\_message\_blob | The received XML message |
| active boolean | True |
| createddate | Filled with date value close to moment of processing the message |
| modifieddate | Filled with date value close to moment of processing the message |
| msl\_soap\_transaction\_id | Filled with transactionid of received message |
| msl\_from\_spr\_code | <sender> |
| msl\_to\_spr\_code | <receiver> |
| msl\_admin | For successful message ‘false’ |

#### A.5 Store data on sending a valid CTRA message to the recipient provider

If recipient serviceprovider supports SOAP: check soap\_xml\_messages and soap\_xml\_transactions.

|  |  |
| --- | --- |
| **database field**  **Soap\_xml\_messages** | **validation** |
| id | No validation |
| sxm\_sender | <sender> |
| sxm\_receiver | <receiver> |
| sxm\_xml\_message\_type | ‘Request answer’ |
| sxm\_xml\_message | A valid XML message [how to check?] |
| sxm\_xml\_message\_datetime | Filled with date value close to moment of processing the message |
| sxm\_status | Depends upon soap listener being able to receive message and return transactionid |
| sxm\_status\_datetime | Depends on soap listener |
| sxm\_retry\_count | Depends on soap listener |
| sxm\_it\_id | Corresponds with soap-version supported by donor |
| sxm\_dt\_id | 2 if <business> = Y, 1 in other cases. |
| active | 'true' |
| createddate | Filled with date value close to moment of processing the message |
| modifieddate | Filled with date value close to moment of processing the message |

|  |  |
| --- | --- |
| **database field**  **Soap\_xml\_transactions** | **validation** |
| id | No validation |
| message\_id | refers to soap\_xml\_message record |
| retry\_count | Depends on soap listener |
| sxm\_sender | <recipient> |

#### A.6 Store a valid CTP message

Not developed

|  |  |
| --- | --- |
|  |  |
| Test purpose | Verify that a valid CTP that is sent to COMP is stored in the database as designed |
| Test steps | Send CTR message  Send CTRA OK message  Send CTP message after ctd\_requesteddatetime |
| Conditions | * request is of the correct soap version for the recipient operator; * DossierID of messages is identical |
| Variations: | Valid CTP messages, variations:   * Different recipient operators: v2, v3, with or without GUI, GUI only * V2 and V3; * Different donor operators: v2, v3, with or without GUI, GUI only. * Donor with and without business\_gui\_only * With and without note tag present * Before wishdatetime * After wishdatetime * With and without messageid; * With and without notefield |

<database to be detailed>

#### A.7 Store data when sending a valid CTP message to the donor provider

Not developed:

|  |  |
| --- | --- |
|  |  |
| Test purpose | Verify that a valid CTP that is sent to COMP is forwarded correctly to the donor provider. |
| Test steps | Send CTR message  Send CTRA OK message  Send CTP message after ctd\_requesteddatetime |
| Conditions | * request is of the correct soap version for the recipient operator; * DossierID of messages is identical |
| Variations: | Identical to variations in Store a valid CTP message |

<database to be detailed>

#### A.8 Store a valid CTC message

<not developed>

|  |  |
| --- | --- |
|  |  |
| Test purpose | Verify that a valid CTC that is sent to COMP is stored in the database as designed |
| Test steps | Send CTR message  Send CTRA OK message  Send CTC message |
| Conditions | * request is of the correct soap version for the recipient operator; * DossierID of messages is identical |
| Variations: | Valid CTC messages, variations are in service providers:   * Different recipient service providers: v2, v3, with or without GUI, GUI only * V2 and V3; * Different donor providers: v2, v3, with or without GUI, GUI only. * Donor with and without business\_gui\_only * With and without note tag present * Before wishdatetime * After wishdatetime * With and without messageid; * With and without notefield |

<database to be detailed>

#### A.9 Store data when send a valid CTC message to the donor provider

Not developed:

|  |  |
| --- | --- |
|  |  |
| Test purpose | Verify that a valid CC that is sent to COMP is forwarded correctly to the donor provider. |
| Test steps | Send CTR message  Send CTRA OK message |
| Conditions | * request is of the correct soap version for the recipient operator; * DossierID of messages is identical |
| Variations: | Identical to variations in Store a valid CTC message |

<database to be detailed>

#### A.10 Store a valid CTCH message

Not developed:

|  |  |
| --- | --- |
|  |  |
| Test purpose | Verify that a valid CTCH that is sent to COMP is stored in the database as designed |
| Test steps | Send CTR message  Send CTRA OK message  Send CTCH message |
| Conditions | * request is of the correct soap version for the recipient operator; * DossierID of messages is identical * CTD-record is present with ctd\_dossier\_status = ‘Pending’ and ctd\_process\_status = ‘Request Answer OK’, ‘Change Answer OK’ or ‘Change Answer Blocking’. * existing and active recipient and donor provider * dossierId is equal to ctd\_switchid * Wishdatetime is not equal to ctd\_ctd\_requesteddatetime; * Wishdatetime is not more than par\_min\_request\_date after the current date and not larger than par\_max\_end\_date (or par\_business\_max\_request\_date if ctd\_ctr\_dt\_id = 2). |
| Variations: | Valid CTCH messages, variations:   * Different types of recipient provider: v2, v3, with or without GUI, GUI only * V2 and V3 messages if appropriate * Different donor provider: v2, v3, with or without GUI, GUI only. * Donors with and without business\_gui\_only * With and without note tag present * With and without messageid; * With and without timestamp |

<database to be detailed>

#### A.11 Store data when sending a valid CTCH message to the donor provider

Not developed

|  |  |
| --- | --- |
|  |  |
| Test purpose | Verify that a valid CTCH that is sent to COMP is forwarded to the designated receiver |
| Test steps | Send CTR, CTRA OK message as preparation;  Send CTCH |
| Conditions | Identical to “Store a valid CTCH message”;   * Receiver must receive CTCH message in SOAP. |
| Variations: | Identical to variations in Store a valid CTCH message |

<database checks to be detailed>

#### A.12 Store a valid CTCHA message

Not developed:

|  |  |
| --- | --- |
|  |  |
| Test purpose | Verify that a valid CTCHA that is sent to COMP is stored in the database as designed |
| Test steps | Send CTR message  Send CTRA OK message  Send CTCH message  Send CTCHA message |
| Conditions | * request is of the correct soap version for the donor operator; * DossierID of messages is identical |
| Variations: | Valid CTCHA messages, variations:   * Different recipient providers: v2, v3, with or without GUI, GUI only. * Different donor providers: v2, v3, with or without GUI, GUI only. * Recipient provider with and without business\_gui\_only * With and without note tag present * With and without messageid; * With and without timestamp * Accepting: blocking = N and blocking: blocking = Y; * With valid wishdatetime (if blocking = N) * With valid proposeddatetime (if blocking = Y) |

[database to be detailed]

#### A.13 Send a valid CTCHA message to the recipient service provider

Not developed

|  |  |
| --- | --- |
|  |  |
| Test purpose | Verify that a valid CTCH that is sent to COMP is forwarded to the designated receiver |
| Test steps | Send CTR, CTRA OK message as preparation;  Send CTCH |
| Conditions | Identical to “Store a valid CTCH message”;   * Receiver must receive CTCH message in SOAP. |
| Variations: | Identical to variations in Store a valid CTCH message |

<Database to be detailed>

### Testing smart validations (sunny day)

Already built in SOAPui, only for SOAP recipient. This should also be developed for GUI.

|  |  |
| --- | --- |
|  |  |
| Test purpose | Verify that the smart validations work as designed. This means that on a CTR message one of the following combination of fields to identify the customer is present:   |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | Field Options -> | 1 | 2 | 3 | 4 | 5 | 6 | 7 | | < postcode> |  |  |  |  |  |  |  | | <housenr> |  |  |  |  |  |  |  | | <contractid> |  |  |  |  |  |  |  | | <IBAN> |  |  |  |  |  |  |  | | <numberseries start> |  |  |  |  |  |  |  | |
| Test steps | 1. Send CTR with the validation fields in the testcase description, store transactionId 2. Check soap\_transactions table for successful “Received” record with the stored transactionId; 3. Wait for poller; 4. Check for raw\_soap\_xml\_record with relation to soap\_transaction record and with state “Done” 5. Check for message\_logging record with the clc\_id for processed OK (57 on test) 6. Check if CTD-record is created with    1. ctd\_switchid = message <dossierid>,    2. ctd\_process\_status = ‘Request’,    3. ctd\_dossier\_status = ‘Pending’   Note: the built testcases also send a CTRA, this is to close the dossier in the test environment and not necessary for this test.  Note 2: the test suite is currently built on the assumption that par\_contract\_id\_business\_dossier = TRUE and will skip testcases for business requests that do not contain a contractid. This is the current situation in production, but also needs to be tested for par\_contract\_id\_business\_dossier = TRUE. |
| Conditions | * request is of the correct soap version for the recipient operator; * Unique dossierId * Unique contracted/telephone number/iban for postcode; * for soap v2 and soapv3 messages with the flag <business>N</business> * if parameters par\_contract\_id\_business\_dossier = FALSE then also for soapv3 messages with the flag <business>Y</business> |
| Variations | * Different types of recipient provider (GUI/SOAPv2/SOAPv3, business\_gui\_only); * Different type of messages: business Y and N; * Different type of donor provider(GUI/SOAPv2/SOAPv3, business\_gui\_only); |

#### Smart Validation test cases

|  |  |
| --- | --- |
| **Number** | **description** |
| A4-1. | Only contractid |
| A4-2. | Only iban |
| A4-3. | Only telephonenumber start |
| A4-4. | Telephonenumberstart = telephonenumberend |
| A4-5. | Contractid + iban |
| A4-6. | Contractid + telephonenumberstart |
| A4-7. | Contractid + telephonenumberstart + telephonenumberend |
| A4-8. | Iban + telephonenumberstart |
| A4-9. | iban + telephonenumberstart + telephonenumberend |
| A4-10. | all |

### Testing proces flow validation state transitions (sunny day)

Developed in SOAPui

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | |  | |
| Test purpose |  | | Verify that all state transitions are done correctly. | |
| **Nr** | **State transitions** | | **Messages to be send** |
| B2.1 | Blocking | | CTR / CTRA blocking |
| B2.2 | Accepted, performed directly | | CTR / CTRA OK / CTP (after wishdatetime) |
| B2.3 | Accepted, performed buffered | | CTR / CTRA OK / CTP (before wishdatetime) |
| B2.4 | Accepted, cancelled | | CTR / CTRA OK / CTC |
| B2.5 | Accepted, Change, performed directly | | CTR / CTRA OK / CTCH / CTP (after wishdatetime) |
| B2.6 | Accepted, Change, performed buffered | | CTR / CTRA OK / CTCH / CTP (before wishdatetime) |
| B2.7 | Accepted, Change, cancelled | | CTR / CTRA OK / CTCH / CTC |
| B2.8 | Accepted, Change, Change OK, performed directly | | CTR / CTRA OK / CTCH / CTCHA OK / CTP (after wishdatetime) |
| B2.9 | Accepted, Change, Change OK, performed buffered | | CTR / CTRA OK / CTCH / CTCHA OK / CTP (before wishdatetime) |
| B2.10 | Accepted, Change, Change OK, cancelled | | CTR / CTRA OK / CTCH / CTCHA OK / CTC |
| B2.11 | Accepted, Change, Change blocking, performed directly | | CTR / CTRA OK / CTCH / CTCHA blocking / CTP (after wishdatetime) |
| B2.12 | Accepted, Change, Change blocking, performed buffered | | CTR / CTRA BLOCKING / CTCH / CTCHA blocking / CTP (before wishdatetime) |
| B2.13 | Accepted, Change, Change blocking, cancelled | | CTR / CTRA BLOCKING / CTCH / CTCHA blocking / CTC |
| B2.14 | Accepted, Change, Change blocking, Change, Change OK, Performed buffered | | CTR / CTRA OK / CTCH / CTCHA blocking / CTCH / CTCHA OK / CTP (after wishdatetime) |
| B2.15 | Accepted, Change, Change blocking, Change, Change OK, Performed directly | | CTR / CTRA OK / CTCH / CTCHA blocking / CTCH / CTCHA OK / CTP (before wishdatetime) |
| B2.16 | Accepted, Change, Change blocking, Change, Change OK, Canceled | | CTR / CTRA OK / CTCH / CTCHA blocking / CTCH / CTCHA OK / CTC |
| B2.17 | Accepted, Change, Change OK, Change, Change blocking, Performed buffered | | CTR / CTRA OK / CTCH / CTCHA OK / CTCH / CTCHA blocking / CTP (after wishdatetime) |
| B2.18 | Accepted, Change, Change OK, Change, Change blocking, Performed directly | | CTR / CTRA OK / CTCH / CTCHA OK / CTCH / CTCHA blocking / CTP (before wishdatetime) |
| B2.19 | Accepted, Change, Change OK, Change, Change blocking, Canceled | | CTR / CTRA OK / CTCH / CTCHA OK / CTCH / CTCHA blocking / CTC |
| Test steps |  | | * Send message(s) in the prescribed order; * After each message check if CTD status is as expected * Check that the message is forwarded or not to the receiver as expected. | |
| Conditions |  | | * Correct messages; * Message processing has been tested successfully | |
| Variations: |  | | * V2/V3; * Different donors: V2, V3, * with/without GUI, * business\_gui\_only Yes/No * business = Y, business = N * All allowed blocking codes for situation | |

### Testing duplicate messages (rainy day)

Developed in SOAPUI

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | |  | |
| Test purpose |  | | Verify that a duplicate message is rejected with the correct error code. | |
| **Nr** | **Dossier status** | | **Messages to be tested for rejection** |
| B3.1 | Request | | CTR with identical dossierid |
| B3.2 | Request Answer OK | | CTRA OK, CTRA blocking |
| B3.3 | Request Answer Blocking | | CTRA OK, CTRA blocking |
| B3.4 | Change | | CTCH |
| B3.5 | Change Answer OK | | CTCHA OK, CTCHA-blocking |
| B3.6 | Change Answer blocking | | CTCHA OK, CTCHA-blocking |
| B3.7 | Cancelled | | CTC |
| B3.8 | Pending/Performed | | CTP |
| B3.9 | Completed/Performed | | CTP |
| Test steps |  | | * Send message(s) to create a dossier in the required status; * Send the messages to be tested for rejection; * Check that the message is rejected * Test subject is the state transition for the recipient. the CTRA and CTCHA messages are sent by the donor. | |
| Conditions |  | | * SOAPv2, SOAPv3 * business = Y and N * Donor of different types; * With and without GUI setup; * With and without business\_gui\_only. | |
| Variations: |  | | * The testcases are performed per type of provider. For the tests for sending CTRA and CTCHA the "Donor" provider should send the CTR so the provider under test can send the CTRA. | |

### Testing Error Codes (rainy days)

Already built in SOAPUI

|  |  |
| --- | --- |
| B4.1 | unknown sender SP code -220 |
| B4.2 | unknown receiver SP code -221 |
| B4.3 | unknown recipient SP code -220 |
| B4.4 | unknown donor SP code -221 |
| B4.5 | inactive recipient -701 / donor 209 (startup script makes serviceprovider inactive, teardown script enables service provider). |
| B4.6 | inactive donor - 210 |
| B4.7 | wishdate = maxwishdate + 1 - blocking 8 |
| B4.8 | wishdate = minwishdate - 1 - blocking 9 |
| B4.9 | donor in subject could not be found - 216 |
| B4.10 | ContractID already exists - blocking 1 |
| B4.11 | no contractid, iban and telephone number - blocking 10 |
| B4.12 | CTRA wishdate <> CTR wishdate - 231 |
| B4.13 | COMP only blockingcode used - 235 |
| B4.14 | Business only blockingcode used on non-business request - 236 |
| B4.15 | invalid blockings - 99 without note - 232 |
| B4.16 | invalid blockings - non exisiting blocking code- 233 |
| B4.17 | invalid blockings - inactive blocking code - 234 |
| B4.18 | invalid blockings – COMP only blocking code- 235 |
| B4.19 | invalid blockings – business only blocking code- 236 |
| B4.20 | CTCHA with wrong dates - wishdate <> CTCH wishdate - 238 |
| B4.21 | CTCHA with wrong dates – wishdate present on CTCHA blocking - 751 |
| B4.22 | CTCHA with wrong dates – proposeddate present on CTCHA OK - 751 |
| B4.23 | CTCHA with wrong dates – blocking with proposeddate in the past - 230 |
| B4.24 | CTCHA with wrong dates – blocking with proposeddate > max\_days – 503???? (is allowed) |
| B4.25 | CTCHA blocking with no dates – 751 |
| B4.26 | CTCHA ok with no dates - 751 |
| B4.27 | Inactive interface CTR - 222 |
| B4.28 | CTR invalid switchid - 216 [check if duplicate of B4.9] |
| B4.29 | CTRA proposeddate in the past – 230 (too far in the future is allowed) |
| B4.30 | unknown sender on SOAP – SOAP 700 |
| B4.31 | answers to inactive recipient -209 |
| B4.32 | CTCH with wrong dates - wishdate in the past - 230 |
| B4.33 | CTCH with wrong dates – wishdate too far in the future – automatic CTCHA |

### Error codes – unknown how to test

|  |  |
| --- | --- |
| B5.1 | 702 "The processing of the received SOAP message went wrong" |
| B5.2 | 703 "Generating SOAP Transaction ID failed" |
| B5.3 | 704 "Sending of the SOAP ack failed" |
| B5.4 | 705 "Creating a SOAP message from DB failed" |
| B5.5 | 706 "The SOAP xml messages tried to be resend has been exceeding the amount of times the COMP administrator has configured" |
| B5.6 | 707 "Sending a SOAP Message failed" |
| B5.7 | 710 "Storing the data to the DB failed" |
| B5.8 | 750 – there are some testcases, but is it complete? |

### Scheduled processes

Testcases need to be developed – automation possible or manual check?

|  |  |
| --- | --- |
| B6.1 | Sending CTP for Hybrid Performed Support; |
| B6.2 | Sending buffered Performed |
| B6.3 | Automatic cancelling; |
| B6.4 | Status emails; |
| B6.5 | Batch notifications |
| B6.6 | Automatic closing dossiers |

## Functional tests COMP sends SOAP message

The sending of outgoing SOAP messages is done by:

* Ensure correct settings for donor that should receive messages;
* Setup SOAP listener (SOAPUI mock service) or other;
* Ensure connectivity between COMP server(s) and SOAP listener;

Note: the soap-version is tested in the process flow validations / state transitions.

Test steps generic for all messages

|  |  |
| --- | --- |
| B7.1 | * Make sure entries for valid XML messages with the configured SOAP-listener as receiver are in the soap\_xml\_messages table. * Ensure the correct sender and receiver are used in the header; * Ensure the timestamp and messageid are present in the header; * Ensure that the dossierid in the message body is correct; * Ensure that the note field in the message body is forwarded correctly |

Specific teststeps for CTR

|  |  |
| --- | --- |
| B7.2 | * Ensure that in a CTR the donor and recipient fields are filled correctly; * Ensure that the dossierid is correct; * Ensure that all validationfields (postcode, housenumber, housenumber\_ext, iban, contractid and/or telephone number start + end) are forwarded correctly; * Ensure that the wishdatetime is forwarded unchanged; * Ensure that the customer name is correct; |

Specific tests for CTRA:

|  |  |
| --- | --- |
| B7.3 | * Ensure the blocking and blockingcode are correct; * Ensure the wishdatetime and/or proposedatetime are correct; * Ensure the blocking and blockingcode are correct; * Ensure the wishdatetime and/or proposedatetime are correct; |

Specific tests for CTCH:

|  |  |
| --- | --- |
| B7.4 | * Ensure the wishdatetime is correct; |

Specific tests for CTCHA

|  |  |
| --- | --- |
| B7.5 | * Ensure the blocking is correct; * Ensure the proposeddatetime or wishdatetime is correct; |

Specific tests for CTP

|  |  |
| --- | --- |
| B7.6 | * Ensure the actualdatetime is correct; |

There are no specific tests for CTC.

Variations:

* Different SOAP versions for sender and receiver;
* Sender is GUI;
* (if it is possible to receive the messages that proves that the sending works);
* Verify that retry mechanism works.

## Functional tests GUI

Testcases need to be developed – can Selenium be used?

* All screens must be tested;
* All error/warning conditions must be tested
* All message processing / state transitions must be tested where recipient = GUI and donor = GUI. (GUI/soap, soap/GUI. GUI/GUI).
  1. GUI Only, SOAPv2 + GUI, SOAPv3 + GUI, SOAPv2 + SOAPv3 + GUI;
  2. Business GUI only;
  3. Business = Y and Business = N;
  4. All allowed variations of smart validations;
  5. With/without housenr\_ext, notefield, numberseries\_end;
* Force sending CTP’s;
* Resending messages.
* GUI messages (not CTR) send from Dossier Details screen and appropriate status screen.

### Functionality tests

To be developed:

* Test ContractId regular expression provider with one regular expression: sunny day (CTR accepted);
* Test ContractId regular expression provider with one regular expression rainy day (CTRA blocking sent)
* Test ContractId regular expression provider with multiple regular expressions (separated by ‘%’): sunny day (CTR accepted) for each regular expression;
* Test ContractId regular expression provider with multiple regular expressions (separated by ‘%’): rainy day (CTRA blocking sent) for each regular expression;

Variations:

* Using SOAP and GUI for recipient operator;
* Ensure that COMP validates against regular expressions of the donor provider, not the recipient provider.

### State transitions between GUI and SOAP

SOAPUI project is created where GUI interaction is manual. Integration possible between SOAPUI and Selenium?

* All scenarios from 3.1.4 where recipient uses soap and donor uses GUI;
* All scenarios from 3.1.4 where recipient uses GUI and donor uses SOAP;

These should include:

* Providers with GUI only, SOAPv2 + GUI, SOAPv3 + GUI, SOAPv2 + SOAPv3 + GUI;
* Providers with business\_gui\_only = true and false.
* CTR’s with business=Y and business=N.

Current SOAPUI has two testSuites:

Donor is GUI and Recipient is GUI. The GUI interaction is controlled by pop-ups (manual test steps) in SOAPUI that ask the test to perfrom certain tasks in the GUI.

### Testing settings

Testcases need to be developed for testing if the following parameters work, both for GUI and for SOAP.

|  |  |
| --- | --- |
| Par\_ctp\_wishdate\_margin | is used for hybrid performed sending. This sets the number of days the CTP message is sent after the wishdatetime in a scenario where the donor supports the CTP message and the recipient does not. Will become obsolete soon. |
| Par\_max\_end\_date | see below. |
| Par\_min\_request\_date | see below. |
| Par\_password\_reset\_days | will be tested manually |
| Par\_archive\_period\_months | will be tested manually |
| Par\_gui\_refresh\_timeframe; | will be tested manually |
| Par\_ctra\_blocking99\_check; | Production setting is TRUE. Test if note field is mandatory with setting is true, change setting to false, send CTR and CTRA blocking 99 without note field, check CTRA is accepted and dossier is blocked. |
| Par\_soap\_retry\_notification; | will be tested manually |
| Par\_numberrange\_check; | Production setting is TRUE: the phone number must be validated against the regex, otherwise not. Set to FALSE, send CTR with phonenumber that does not conform to regex, CTR must be accepted. Set to TRUE, send CTR with phonenumber that does not conform to regex, CTR must be rejected. |
| Par\_numberrange\_regex; | set par\_numberrange\_check to TRUE. Send CTR's with telephonenumber start conforming and non-conforming numbers and check for acceptance/rejection. Try different regex-es. |
| Soap timeframes; | will be tested manually |
| Par\_soap\_max\_message\_incoming\_packet | will always be 1, no test necessary. |
| Par\_soap\_max\_message\_outgoing\_packet | will always be 1, no test necessary |
| Par\_ctp\_buffering | Production setting is TRUE. For test: send CTR with wishdate in the future, send CTRA OK, send CTP, check ctd statuses are Pending/Performed and no Performed message is sent. Set setting to FALSE. send CTR with wishdate in the future, send CTRA OK, send CTP, check ctd statuses are Completed/Performed and that Performed message is sent. |
| Par\_business\_max\_request\_date | see below. |
| Par\_contract\_id\_business\_dossier; |  |
| Par\_admin\_logging\_buffering. |  |

#### Testing max wishdate (sunny day)

Normal production setting is 90. Try with setting = 90, also with higher and lower values.

CTR with wishdatetime = todayT23:59

CTR with wishdatetime = max. days + today;

CTRA blocking proposeddatetime on limits

CTCH with wishdatetime = todayT23:59

CTCH with wishdatetime = max. days + today;

CTCHA blocking with proposeddatetime on limits.

#### Testing max wishdate (rainy day)

Normal production setting is 90. Try with setting = 90, also with higher and lower values.

CTR with wishdatetime = todayT00:00

CTR with wishdatetime = yesterday;

CTR with wishdatetime = max. days + today + 1;

CTRA blocking proposeddatetime over limitations

CTCH with wishdatetime = todayT00:00

CTCH with wishdatetime = yesterday;

CTCH with wishdatetime = max. days + today + 1;

CTCHA with proposeddatetime over limitations.

#### Testing max business wishdate (sunny day)

Normal production setting is 120. Try with setting = 120, also with higher and lower values.

CTR with <business>Y</business> and wishdatetime = todayT23:59

CTR with wishdatetime = max. days + today;

CTRA blocking proposeddatetime on limits

CTCH with wishdatetime = todayT23:59

CTCH with wishdatetime = max. days + today;

CTCHA blocking with proposeddatetime on limits.

#### Testing max business wishdate (rainy day)

Normal production setting is 120. Try with setting = 120, also with higher and lower values.

CTR with <business>Y</business> and wishdatetime = todayT00:00

CTR with <business>Y</business> and wishdatetime = yesterday;

CTR with <business>Y</business> and wishdatetime = max. bus.days + today + 1;

CTRA blocking proposeddatetime over limitations

CTCH with wishdatetime = todayT00:00

CTCH with wishdatetime = yesterday;

CTCH with wishdatetime = max. days + today + 1;

CTCHA with proposeddatetime over limitations.

## Performance tests

Run one of the soap-state transition sunny day scenarios with SOAPui as loadtest. This should be done with an appropriate number of total scenarios and threads.

## Testing notification emails, reporting emails and error emails

Must be developed.

* Verify that periodically emails are sent to report dossiers that require action
* Verify that periodically emails are sent to announce automatic cancellation of messages;
* Verify that emails are sent for GUI only/business GUI only providers when messages are sent to them;
* Verify that emails are sent when errors occur on the processing of a message.